УДК 519.6 DOI: 10.18698/0236-3933-2022-2-42-59

СТРУКТУРНОЕ СОГЛАСОВАНИЕ АЛГОРИТМОВ ГЛОБАЛЬНОЙ ОПТИМИЗАЦИИ С АРХИТЕКТУРОЙ ГРАФИЧЕСКИХ ПРОЦЕССОРОВ

Е.Ю. Селиверстов evgeny.seliverstov@omniverse.ru

МГТУ им. Н.Э. Баумана, Москва, Российская Федерация

Аннотация

Использование современных метаэвристических алгоритмов глобальной оптимизации требует применения параллельных вычислительных систем, в частности графических процессорных устройств со сложной архитектурой. Параллельная эффективность алгоритма оптимизации сильно зависит от выбранного отображения алгоритма на вычислительную систему. Предложен алгоритм согласования параллельного метаэвристического оптимизационного алгоритма с архитектурой графических процессорных устройств. Рассмотрена постановка задачи согласования как задачи определения оптимальной стратегии параметризованного алгоритма и оптимального отображения этого алгоритма на архитектуру вычислительной системы. Предложено математическое представление графовых отображений и ограничений отображений. Отображение алгоритма на вычислительную систему представлено как отображение графовой параметризованной модели параллельного алгоритма с островной моделью параллелизма на совокупность графовых моделей графического процессора. Множество допустимых отображений на основе указанных ограничений формализует особенности архитектуры вычислительной системы и модели параллелизма алгоритма. Предложен алгоритм структурного согласования, основанный на совместном решении задачи оптимизации и задачи структурной оптимизации допустимых отображений. Рассмотрен критерий параллельной эффективности, оцениваемый как экспериментально, так и аналитически по графовым моделям и их отображениям, что позволяет проводить оптимизацию для различных сценариев поиска отобра-

Ключевые слова

Глобальная оптимизация, задача согласования, задача отображения, структурная оптимизация, параллельные системы, графические процессоры

жений. Приведены результаты вычислительного эксперимента со сравнением параллельной эффективности параллельного алгоритма, основанного на алгоритме структурного согласования, и классического алгоритма для класса тестовых задач оптимизации из пакета CEC (Congress of Evolutionary Computing)

Поступила 10.03.2022
Принята 31.03.2022

Введение. Применение метаэвристических алгоритмов глобальной оптимизации, которые отличаются высокой вычислительной сложностью и размерностью, требует использования параллельных вычислительных систем, например, высокопроизводительных графических процессорных устройств (ГПУ). Архитектура ГПУ значительно отличается от архитектур классических многоядерных систем. Отличительной особенностью является векторная SIMD-структура на нижнем уровне и MIMD-структура на верхнем уровне, иерархическая структура памяти, простота и большое число вычислительных ядер [1].

В основе задачи согласования алгоритмов с архитектурой параллельных вычислительных систем лежит задача отображения взаимодействующих вычислительных задач (task mapping, task scheduling) [2] на вычислительную систему. Для решения этой задачи предложены несколько аналитических метрик (время вычислений и коммуникаций) и способы их оценки по графу задач, диаграмме Ганта [3]. Для ГПУ важными метриками являются утилизация мультипроцессоров, латентность при обращении к различным классам памяти. Известны подходы к отображению графов задач на вычислительную систему. Модели двудольных графов и гиперграфов, классические методы разбиения графов на основе минимизации разреза потока, методы иерархического разбиения графов рассмотрены в [4]. В классификации методов статического планирования [5] выделяют машинно-зависимые графы задач и абстрактные графы вычислительных задач, а также методы их отображения на графы вычислительной системы путем кластеризации вершин графов. Оптимизационный подход к задаче отображения рассмотрен в [6]. Сначала осуществляют минимизацию метрики скалярного отклонения между графом задач и системой. Затем решают задачу структурной оптимизации графов. Для перечисления допустимых отображений применяются методы порождения семейств графов на основании формальных грамматик. Методы отображения для ГПУ основаны преимущественно на динамическом планировании независимых или слабо связанных графов задач и на локальных очередях задач, что позволяет достигать высокой параллельной эффективно-

сти для задач определенного класса [7]. Графы потока управления параллельных метаэвристических алгоритмов относят к графам взаимодействующих задач, а исследования для отображения таких графов на ГПУ практически отсутствуют.

Следовательно, алгоритмы согласования, учитывающие особенности архитектуры ГПУ и параметризованных графов задач, развиты слабо. Исследования методов отображения алгоритмов оптимизации проводятся преимущественно для конкретных алгоритмов оптимизации. *Цель исследования* — разработка методов и алгоритмов решения задачи структурного согласования метаэвристических алгоритмов глобальной оптимизации и архитектуры ГПУ, основанных на графовой модели класса алгоритмов оптимизации и графовой модели ГПУ.

Модели параллельного алгоритма и параллельной вычислительной системы. Назовем исходную задачу глобальной оптимизации базовой задачей оптимизации. Метаэвристический алгоритм, используемый для решения базовой задачи, назовем базовым алгоритмом оптимизации. Определим базовую задачу оптимизации в виде

$$\min_{X \in D_X} \Phi(X) = \Phi(X^*),$$

где $\Phi(X)$ — целевая функция задачи; X^* — оптимальное значение вектора параметров базовой задачи; D_X — область допустимых значений вектора X варьируемых параметров.

Рассмотрим параметризованную модель $GA(A, P)$ параллельного алгоритма метаэвристической оптимизации $A(P)$ с островной моделью параллелизма [8, 9] в виде ориентированного графа $GA = (VA, EA)$, где множество вершин VA представляет собой операции алгоритма; множество ребер EA — зависимости по данным между операциями [10]. Вектор свободных параметров P алгоритма A состоит из параметров, представляющих свойства алгоритма оптимизации (значения свободных параметров) и островной модели параллелизма (число островов N_S, размер острова N_P). В графе GA выделим отдельные подграфы PD_i, $i \in [0;\ N_S)$, соответствующие операциям оптимизации последовательными алгоритмами над независимыми субпопуляциями (островами) I_i, $i \in [0;\ N_S)$; подграфы PE_i и PR_i, соответствующие операциям коммуникации и синхронизации.

Параллельной моделью вычислительной системы M для выполнения алгоритма A является модель блочно-синхронного параллелизма BSP (Bulk Synchronous Parallel) [11], разделяющая выполнение программы

на супершаги, состоящие из фазы независимых вычислений, фазы коммуникации и фазы синхронизации. Модель вычислительной системы состоит из трех взаимосвязанных графовых моделей [12]: структурной модели GS, отражающей иерархическое устройство системы из отдельных вычислительных устройств; коммуникационной модели GC, представляющей коммуникации между вычислительными устройствами; модели памяти GM, определяющей доступность различных видов памяти для вычислительных устройств. Подробно модели алгоритмов и системы, их представление и свойства рассмотрены в [10, 12].

Задача согласования. Представим задачу согласования как задачу определения оптимальной стратегии P параметризованного метаэвристического алгоритма оптимизации $A(P)$ и оптимального отображения алгоритма на модель ГПУ на основании некоторого критерия оптимальности согласования.

Задачу структурно-параметрического согласования ставим как задачу комбинаторной оптимизации в пространстве, формируемом из множеств графовых отображений моделей алгоритма и параллельной системы. Результатом задачи согласования полагаем оптимальное значение вектора управляющих параметров, полученное из решения указанной задачи оптимизации. Задачу согласования TO сводим к задаче установки параметров модели алгоритма и к задаче оптимизации T-отображения, являющейся выраженной многокритериальной многопараметрической задачей оптимизации:

$$\min^{M}_{P \in D_P,\ T \in L_{TP}} \Psi(P, T) = \Psi(P^*, T^*).$$

Здесь $\Psi(P, T)$ — векторный критерий оптимальности. Под $\min^M$ подразумеваем многокритериальную оптимизацию векторного критерия.

Определим отображение одного графа на другой.

Вершинное отображение (или просто отображение) графа G на H обозначим как $\lambda(G, H)$ и представим вектором

$$\lambda = (\lambda_1, \ldots, \lambda_n), \quad n = \lambda,$$

где

$$\lambda_k = (g_k, h_k), \quad k \in [1, ..., n];$$

$$g_k = \{V(G)_i, \quad i \in I_{G,\,k}\}, \quad h_k = \{V(H)_j, \quad j \in I_{H,\,k}\}.$$

Пары $\lambda_k = (g_k, h_k)$ назовем компонентами вершинного отображения. Индексное множество I_G содержит в k-м элементе $I_{G,\,k}$ индексы

вершин графа H, отображаемых на вершину $V(G)_i$. Индексное множество I_H включает в себя в k-м элементе $I_{H,k}$ индексы вершин графа G, отображаемых на вершину $V(H)_j$.

Вершинное отображение составим так, чтобы множества компонентов g_k, h_k образовывали непересекающееся разбиение и были справедливы следующие соотношения:

$$\forall g_k,\ g_m,\ k \neq m : g_k \cap g_m = \varnothing;$$
$$\forall h_k,\ h_m,\ k \neq m : h_k \cap h_m = \varnothing.$$

Определим понятие реберного отображения $\eta(G, H)$ графа G на граф H. Будем утверждать, что ребро $e_k \in E(G)$ отображается на ребро $e_m \in E(H)$ тогда и только тогда, когда начальные вершины ребер G_{1k}, H_{1m} находятся в одной компоненте отображения, а конечные вершины G_{2k}, H_{2m} — в другой. Более формально реберное отображение определим как множество пар ребер, удовлетворяющих условию

$$\eta(G, H)\ \{(e_k, e_m) : e_k \equiv (G_{1k}, G_{2k}),\ e_m \equiv (H_{1m}, H_{2m});$$
$$\exists g_i, h_j, i \neq j : \lambda_i = (g_i, h_i),\ \lambda_j = (h_j, g_j);$$
$$G_{1k} \in g_i,\ H_{1m} \in h_i,\ G_{2k} \in g_j,\ H_{2m} \in h_j\}.$$

Для различных графовых моделей G, H и отображений $\lambda(G, H)$, $\eta(G, H)$ определим частные ограничивающие функции отображения: групповую S_I, вершинную S_V, реберную S_E. По этим функциям определим ограничения, формализующие особенности архитектуры графической системы и алгоритма оптимизации.

Ограничивающая функция $S(\lambda, \eta)$ задает функциональное отображение

$$S : \lambda \times \eta \to \{0, 1\}.$$

Частные ограничивающие функции S_I, S_V, S_E, S_D однозначно зададим вспомогательными ограничивающими функциями, определенными над парами вершин или ребер. Например, частная вершинная ограничивающая функция S_V формализует допустимость отображения вершины одного графа на вершину другого графа:

$$S_V(\lambda) = 1 \Leftrightarrow \forall \lambda_k = (g_k, h_k) \in \lambda,\ \forall v_i \in g_k,\ v_j \in h_k : \dot{S}_V(v_i, v_j) = 1.$$

Для задания ограничивающей функции $S_V(\lambda(G, H))$ достаточно перечислить значения вспомогательной ограничивающей функции $\dot{S}_V$ для всего множества $V(G) \times V(H)$.

Общую ограничивающую функцию $S(\lambda, \eta)$ составим из частных ограничивающих функций:

$$S(\lambda, \eta) = (S_I(\lambda), \ S_V(\lambda), \ S_E(\eta)). \tag{1}$$

Общее ограничение выполнимо при удовлетворении всех ограничений

$$S(\lambda, \eta) = 1 \Leftrightarrow \forall S_k \in S : S_k = 1. \tag{2}$$

Назовем отображение $\lambda(G, H)$ допустимым, если значения частных ограничивающих функций $S_I(\lambda)$, $S_V(\lambda)$, $S_D(\lambda)$ равны единице. Аналогично назовем отображение $\eta(G, H)$ допустимым, если значение частной ограничивающей функции $S_E(\eta)$ равно единице. Назовем пару отображений $\lambda(G, H)$, $\eta(G, H)$ допустимой, если значение общей ограничивающей функции $S(\lambda, \eta)$ равно единице.

Определим отношение параметризованной модели алгоритма $GA(A, P)$ и моделей параллельной вычислительной системы GS, GC, GM. Модель системы не параметризуем и задаем фиксированной.

Для общности постановки задачи модели вычислительной системы объединим в кортеж моделей $\mathcal{M} = (GS, GC, GM)$, а модели алгоритма — в одноэлементный кортеж $\mathcal{A}(P) = (GA(A, P))$. Отображение кортежа модели $\mathcal{A}$ на кортеж модели $\mathcal{M}$ зададим как набор попарных реберных и вершинных отображений графов GA, GS, GC, GM в составе этих кортежей. Определим объединенные отображения моделей λ_T, η_T через введенные ранее вершинные и реберные отображения графов:

$$\lambda_T(\mathcal{A}, \mathcal{M}, S) = \left\{\lambda_k : \forall A_i \in \mathcal{A}, M_j \in \mathcal{M} : \lambda_k\ (A_i, M_j), S(\lambda_k, \varnothing) = 1\right\};$$

$$\eta_T(\mathcal{A}, \mathcal{M}, S) = \ \left\{\eta_k : \forall A_i \in \mathcal{A}, M_j \in \mathcal{M} : \eta_k(A_i, M_j), S(\varnothing, \eta_k) = 1\right\}.$$

Здесь функция S задает общее ограничение отображения (2).

Назовем T-отображением модели алгоритма на вычислительную систему кортеж, состоящий из объединенных отображений моделей λ_T, η_T:

$$T(\mathcal{A}, \mathcal{M}, S) = \left(\lambda_T(\mathcal{A}, \mathcal{M}, S), \ \eta_T(\mathcal{A}, \mathcal{M}, S)\right).$$

Пусть T-отображения составляют множество допустимых отображений

$$L_T(\mathcal{A}, \mathcal{M}) = \left\{T(\mathcal{A}, \mathcal{M}, S^1)\right\}. \tag{3}$$

Мощность множества L_T определяет число возможных T-отображений для заданных моделей. Как правило, при слабых ограничениях S справедливо неравенство $L_T > 1$, т. е. существует несколько отображений λ_T, η_T модели алгоритма $\mathcal{A}$ на модель системы $\mathcal{M}$. Модель $\mathcal{A}$ в объединенных отображениях моделей λ_T, η_T параметризована вектором P, поэтому эти отображения неявно зависят от P. Явной функциональной зависимости от параметра P объединенные отображения моделей не имеют, так как для одного значения P существует не одно T-отображение T, а целое их множество $\{T\} \subset L_T(\mathcal{A}, \mathcal{M})$. Отображение алгоритма на вычислительную систему является однозначным, но не взаимно однозначным, поскольку не ставится цель идентифицировать алгоритм по программе. Кроме того, одному и тому же алгоритму и реализующей его программе могут соответствовать несколько параметризованных моделей $GA(A, P)$. Пусть областью допустимых значений вектора параметров P является множество D_P. Пусть $L_{\mathcal{A}}$ и $L_{\mathcal{M}}$ — множества возможных моделей алгоритма $\mathcal{A}$ и системы $\mathcal{M}$. Множество $L_T(\mathcal{A}, \mathcal{M})$ (3) определяет допустимые T-отображения для фиксированных моделей $\mathcal{A}, \mathcal{M}$. Сформируем для $L_{\mathcal{A}}$ и $L_{\mathcal{M}}$ общее множество T-отображений:

$$\dot{L}_T = \bigcup_{\mathcal{A} \in L_{\mathcal{A}},\ \mathcal{M} \in L_{\mathcal{M}}} L_T(\mathcal{A}, \mathcal{M}).$$

Области $\dot{L}_T$ и D_P не связаны явно, но построение T-отображений по модели алгоритма $\mathcal{A}(P)$ задает неявные ограничения на значения $T \in \dot{L}_T$. Вводим булеву функцию ограничения $f_{PT}(P, T)$, определяющую допустимые сочетания параметров P, T. Функция f_{PT} принимает значение единицы тогда и только тогда, когда T-отображение допустимо при значении вектора параметров P. Строим область допустимых T-отображений:

$$L_{TP} = \left\{T : \forall T \in \dot{L}_T : \left(\forall P \in D_P : f_{PT}(P, T) = 1\right)\right\},\ \ L_{TP} \subset \dot{L}_T.$$

Вводим критерии $E_A(\mathcal{A}(P),\ T(\mathcal{A}, \mathcal{M}))$ эффективности параллельной реализации алгоритма $A(P)$. Экспериментальную оценку E_A^e критерия $E_A(\cdot)$ проводим по результатам выполнения программы $Prog(\mathcal{A}, \mathcal{M})$, сформированной на основании моделей алгоритма и системы. Аналитическую оценку E_A^a осуществляем на основании графовой модели алгоритма и системы без выполнения на реальной вычислительной системе.

Алгоритм решения задачи согласования. Рассмотрим алгоритм AO решения задачи согласования. Алгоритм является итерационным алгоритмом оптимизации, итерация которого совпадает с итерацией параллельного алгоритма A (один шаг модели параллельных вычислений BSP).

Введем множество $\mathcal{H}$ истории отображений. Отображение $T \in \mathcal{H}$ назовем исследованным отображением, а прочие отображения — неисследованными отображениями.

Одну итерацию t алгоритма AO строим по следующей схеме.

Шаг 1. Выбираем начальные значения X^0, P^0.

Шаг 2. Инициализируем $\mathcal{H} = \varnothing$.

Шаг 3. Формируем программу $Prog$ на основе графовых моделей $\mathcal{A}$, $\mathcal{M}$, T-отображения T^t и значения вектора параметров P^t.

Шаг 4. Выполняем один шаг BSP программы $Prog$.

Шаг 4а. Выполняем фрагмент этой программы, реализующий операции $pd_i \in PD$, $i \in [0...PD]$ алгоритма $\mathcal{A}$.

Шаг 4б. Выполняем фрагмент программы, реализующий операции коммуникации $pe \in PE$, $i \in [0...PE]$ и синхронизации $pr_j \in PR$, $j \in [0...PR]$ графовой модели GA.

Шаг 5. Пополняем историю отображений $\mathcal{H}$ кортежем $\left(T^t, E_A^e(P^t, T^t)\right)$, показывающим измеренную параллельную эффективность отображения T^t.

Шаг 6. Решаем подзадачу TO_T с помощью алгоритма AO.

Шаг 6а. Формируем графовую модель алгоритма $\mathcal{A}(P^t)$.

Шаг 6б. Формируем допустимое множество T-отображений $L_{TP}(P^t)$.

Шаг 6в. Решаем подзадачу TO_T оптимизации отображения при фиксированных векторах параметров P^t и текущем значении T^t. Получаем оптимальное для шага t T-отображение T^{t+1}. На основании этого T-отображения перестраиваем программу на следующей итерации.

Шаг 7. На основании значений векторов P^t, T^t и текущего значения X^t решаем задачу параметрической оптимизации вектора P и вычисляем оптимальное значение вектора P_p^{t+1}. Исследование методов решения задачи параметрической оптимизации лежит вне рамок нстоящей работы.

Шаг 8. Если условие завершения алгоритма $\mathcal{A}$ выполнено, то завершаем вычисления. Иначе присваиваем $t = t + 1$ и переходим к шагу 3.

Обозначаем общее число итераций алгоритма как $\hat{t}$. Результатами алгоритма AO являются оптимальная стратегия P^* и оптимальное T-отображение $T^{\hat{t},*}$. Каждой итерации t алгоритма AO соответствует номер глобальной итерации $F_G(t)$, где функция F_G определена структурой алгоритма AO_P. В качестве условия завершения алгоритма AO предлагаем условие достижения максимального номера глобальной итерации $t^{\max}$:

$$F_G(t) < t^{\max}. \tag{4}$$

Алгоритм структурной оптимизации отображения. Сформулируем задачу TO_T как задачу однокритериальной дискретной оптимизации

$$\min_{T \in L_{TP}} F(P', T) = F(P', T^*), \tag{5}$$

где P' — фиксированное значение вектора параметров P; $L_{TP}(\mathcal{A}, \mathcal{M})$ — множество допустимых T-отображений (3); $F(P, T) : T \to \mathbb{R}$ — критерий оптимальности задачи согласования.

Для итерационного решения задачи TO_T необходимо определить множество допустимых T-отображений $L_{TP}(\mathcal{A}, \mathcal{M})$, алгоритм вычисления критерия оптимальности F и алгоритм выбора следующего отображения из этого множества.

Задача определения множества $L_{TP}(\mathcal{A})$ может быть решена различными способами. Простейшим способом является систематическое перечисление T-отображений, удовлетворяющих условию допустимости (1). К эффективным способам решения относим поиск множества подграфов графа системы, изоморфных графу алгоритма, на основе алгоритмов Мессмера и Бунке [13].

Алгоритм вычисления критерия оптимальности. Назовем метрикой инерционности алгоритма AO метрику $d_T(T^t, T^{t+1}) : T \times T \to \mathbb{R}$ для двух последовательных шагов t и $t+1$ алгоритма AO при фиксированной стратегии P'. Примером этой метрики служит константная метрика $\tilde{d}_T(T^t, T^{t+1}) = F_B(\neg(T^t \equiv T^{t+1}))$, где $F_B : \{0, 1\} \to \mathbb{R}$ — функция интерполяции булева выражения во множество $\mathbb{R}$. Возможны и более сложные функции, учитывающие особенности модели системы и алгоритма.

Для учета предыдущих итераций алгоритма AO вводим функцию аппроксимированного критерия параллельной эффективности $F_E(T, T')$ для текущего T и предыдущего T'-отображения. Пусть на выполненном шаге алгоритма AO для отображения T получены экспериментальная

оценка критерия $E_A^e(P', T)$ и аналитическая оценка $E_A^a(P', T)$. Исходим из предположения, что аналитическая оценка времени исполнения по графовой модели пропорциональна реальному времени исполнения. Тогда определим функцию масштабирования в виде линейной функции

$$m(T) = \frac{E_A^e(P', T)}{E_A^a(P', T)}. \tag{6}$$

Рассмотрим два случая вычисления значений функции $F_E(T)$.

1. Значение T является исследованным, т. е. содержится в истории отображений $\mathcal{H}$. Тогда аппроксимированный критерий параллельной эффективности в точности равен соответствующему значению $E_A^e(P', T)$.

2. Значение T является неисследованным. Тогда оценку параллельной эффективности выполняем, исходя из значения аналитического критерия E_A^a эффективности параллельной реализации и значения $m(T)$ функции масштабирования (6), учитывающей оценку E_A^e с предыдущего шага.

Функцию вычисления аппроксимированного критерия параллельной эффективности запишем в виде

$$F_E(T, T') \begin{cases} \dot{E}, & \text{если } \exists\left(\dot{T}, \dot{E}\right) \in \mathcal{H} : \dot{T} = T; \\ E_A^a(P', T)\, m(T'), & \text{иначе.} \end{cases} \tag{7}$$

Метрика инерционности d_T накладывает ограничение на область допустимых значений L_{TP} задачи TO_T. При этом задачу нельзя считать задачей условной оптимизации, так как ограничение L_{TP} зависит от текущего значения переменной T, поэтому рассматриваем TO_T как задачу безусловной оптимизации. Предлагаем метод на основе метода штрафных функций [14]. Запишем функцию

$$F(T) = F_E(T, T') + F_P(T),$$

а штрафную функцию $F_P(T)$ с коэффициентом штрафа α определим как

$$F_P(T) = \alpha d_T(T, T')^2.$$

Алгоритм перехода между отображениями. Пусть на шаге t алгоритма AO выбрано текущее отображение T^t. Определим алгоритм AO_{TNEXT} выбора следующего отображения $T^{t+1} \in L_{TP}$. По аналогии с (7) определим вспомогательную функцию $\tilde{p}(T)$ порога оценки вероятности выбора T-отображения на основании истории отображений $\mathcal{H}$:

$$\tilde{p}(T) = \begin{cases} \beta \dfrac{E^{\max} - \dot{E}}{E^{\max}}, & \text{если } \exists\left(\dot{T}, \dot{E}\right) \in \mathcal{H} : \dot{T} = T; \\ \delta, & \text{иначе.} \end{cases}$$

Здесь β — коэффициент чувствительности к выбору уже исследованных отображений; $E^{\max} = \max\left\{\dot{E} \mid \exists\left(\dot{T}, \dot{E}\right) \in \mathcal{H}\right\}$ — максимальное значение критерия E из истории $\mathcal{H}$.

Определим непрерывную случайную величину ξ_T с равномерным распределением. Введем два условия выбора следующего T-отображения:

$$\xi_T < \tilde{p}(T); \tag{8}$$

$$F(T) < F(T^*). \tag{9}$$

Здесь T^* — лучшее найденное T-отображение. Если отображение является исследованным, то оно будет выбрано с вероятностью, определяемой из истории $\mathcal{H}$ и не превышающей β. Чем меньше значение критерия параллельной эффективности $\dot{E}$ относительно $E^{\max}$, тем больше вероятность выбора этого отображения. Неисследованные отображения будут выбраны с оценкой вероятности $1 - \beta$.

Запишем шаги алгоритма AO_{TNEXT}.

Шаг 1. Полагаем значение счетчика $k = 0$, начальное значение $T^* = T^t$ из отображения с итерации t алгоритма AO.

Шаг 2. Выбираем очередное значение T^k.

Шаг 3. Выбираем значение случайной величины $\xi_T \in \left[0; 1\right]$.

Шаг 4. Если условие (8) не выполнено, переходим к шагу 6.

Шаг 5. Если условие (9) для T^k выполнено, то полагаем $T^* = T^k$.

Шаг 6. Если $k < L_{PT}$, выбираем $k = k + 1$ и переходим к шагу 2. Иначе полагаем новое отображение $T^{k+1} = T^*$ и завершаем алгоритм.

Отличием предлагаемого алгоритма AO_{TNEXT} от схожего алгоритма поиска с запретами является адаптивность, обусловленная введением коэффициента чувствительности к исследованным отображениям. Введение зависимости $\tilde{p}$ от значения отображения T позволяет увеличивать вероятность выбора отображения с меньшим значением фитнесс-функции F. В силу пополнения множества $\mathcal{H}$ истории отображений и стохастичности алгоритма AO_{TNEXT} при последующих вызовах алгоритмом AO_{TNEXT} будут выбраны отображения, отличающиеся от текущего. Важным преимуществом предложенного алгоритма является его параметризация для двух различных сценариев:

– конкретизация отображения — поиск отображения с наилучшим значением фитнесс-функции F;

– исследование отображений — повышение разнообразия допустимых отображений и наполнение истории $\mathcal{H}$.

Вычислительный эксперимент. Проведем вычислительные эксперименты для сравнения эффективности предложенного алгоритма оптимизации PAGOS, реализующего алгоритм согласования AO, и классического алгоритма оптимизации CLASSIC с фиксированной стратегией P и T-отображением на хост-систему. В качестве базовых задач оптимизации применим тестовые целевые функции различных классов из пакета CEC (Congress of Evolutionary Computing) [15]. В качестве последовательного алгоритма оптимизации выберем алгоритм оптимизации роем частиц [16] с параметрами $\chi = 0,7298$, $k = 1$, $\varphi_1 = \varphi_2 = 2,05$, $\gamma = 0,5$, размером роя 20 и глобальной топологией соседства. Используем параллельную систему с ГПУ Tesla K80 (GK210) и ЦПУ Intel Xeon E5-2686.

В эксперименте рассмотрим зависимость ускорения от вычислительной сложности целевой функции и размерности базовой задачи оптимизации. Ускорение S определим как отношение времени работы алгоритмов:

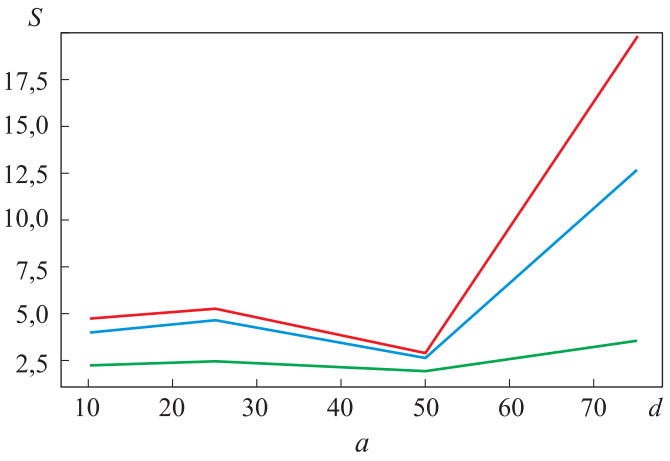
$$S = \frac{\sum\limits_{t=0}^{\hat{t}} E_{E,t}^{\text{CLASSIC}}}{\sum\limits_{t=0}^{\hat{t}} E_{E,t}^{\text{PAGOS}}}.$$

Здесь $E_{E,t}^{Q}$ — значение метрики параллельной эффективности соответствующего алгоритма Q на макрошаге t [17]. Критерием останова алгоритма считаем условие достижения максимального числа итераций $t^{\max}$ (4).

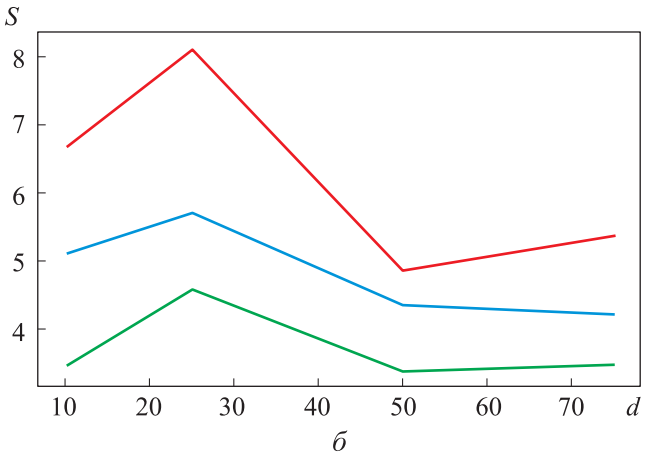
В эксперименте используем следующие параметры алгоритмов: $t^{\max} = 16$, $N_T^0 = 5$, $N_T^{\max} = 5$, $N_S = 64$, $t^{\max} = 50$. Введем масштаб целевой функции *scale*, задающий число повторений расчетов целевой функции.

Графики зависимости ускорения S от размерности d базовой задачи оптимизации для различных целевых функций и значений их масштабов $scale = 500, 1000, 2000$ приведены на рисунке.

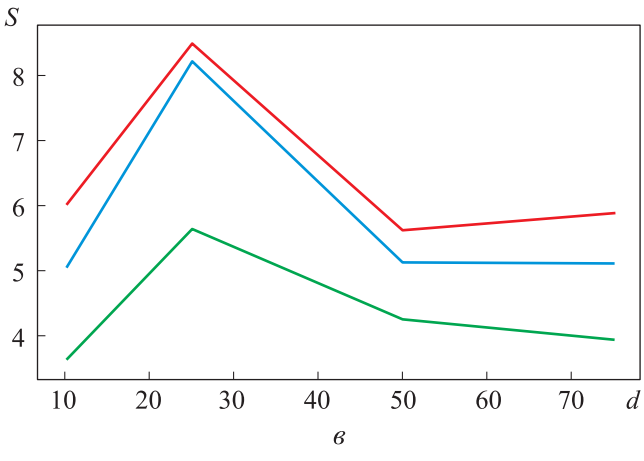
Согласно результатам эксперимента, применение алгоритма PAGOS позволяет сократить общее время решения задачи оптимизации от 3 до 20 раз. Ускорение S растет линейно при увеличении вычислительной сложности целевой функции, что демонстрирует хорошую масштабируемость параллельного алгоритма.



а



б



в

Зависимости ускорения S от размерности задачи d для целевых функций Rastrigin (*а*), Rosenbrock (*б*) и Ackley (*в*) и их масштабов:
scale = 500 (зеленая прямая), 1000 (голубая), 2000 (красная)

Средний размер истории исследованных отображений $|\mathcal{H}|$ составляет 8 для задач Rosenbrock и Rastrigin и 5 для задачи Ackley при константах $\alpha = 0,05$, $\beta = 0,9$, $\delta = 0,9$, соответствующих сценарию конкретизации отображения.

Для мультимодальных задач оптимизации (Rastrigin, Ackley) наибольшее ускорение наблюдается при размерности базовой задачи оптимизации $d = 25$. Зависимости ускорения от размерности базовой задачи схожи. Для унимодальных задач (Rosenbrock) демонстрируется наибольшее ускорение $S = 20$ при размерности задачи $d = 75$.

Заключение. Предложен алгоритм структурно-параметрического согласования метаэвристического оптимизационного алгоритма с архитектурой графических процессорных устройств, основанный на совместном решении задачи метаэвристической оптимизации с островной моделью параллелизма и задачи структурной оптимизации допустимых отображений. Рассмотрена формальная постановка задачи согласования на основании графовой параметризованной модели алгоритма, моделей графического процессора, графовых отображений и ограничений отображений, формализующих особенности архитектуры. Предложенный алгоритм отличается использованием адаптации вектора параметров алгоритма и оптимизации T-отображения во время решения базовой задачи оптимизации. Рассмотрено использование алгоритма согласования для различных сценариев: конкретизация оптимального отображения и исследование отображений для наполнения истории отображений. По результатам вычислительного эксперимента продемонстрировано преимущество алгоритма согласования по параллельной эффективности для ряда тестовых задач. Дальнейшим направлением исследований является развитие метода структурной оптимизации и автоматический синтез ограничений отображений для класса вычислительных систем на основе графических процессоров.

ЛИТЕРАТУРА

[1] Nvidia Ampere GA102 GPU architecture. 2020. *nvidia.com: веб-сайт.* URL: https://www.nvidia.com/content/PDF/nvidia-ampere-ga-102-gpu-architecture-whitepaper-v2.pdf (дата обращения: 10.09.2021).

[2] Kwok Y.K., Ahmad I. Static scheduling algorithms for allocating directed task graphs to multiprocessors. *ACM Comput. Surv.*, 1999, vol. 31, no. 4, pp. 406–471. DOI: https://doi.org/10.1145/344588.344618

[3] Sinnen O. Task scheduling for parallel systems. New York, Wiley-Interscience, 2007.

[4] Hendrickson B., Kolda T.G. Graph partitioning models for parallel computing. *Parallel Comput.*, 2000, vol. 26, no. 12, pp. 1519–1534.
DOI: https://doi.org/10.1016/S0167-8191(00)00048-X

[5] Chen S., Eshaghian M.M., Wu Y.C. Mapping arbitrary non-uniform task graphs onto arbitrary non-uniform system graphs. *Proc. Int. Conf. on Parallel Processing*, 1995, vol. 2, pp. 191–195.

[6] Berman F., Snyder L. On mapping parallel algorithms into parallel architectures. *J. Parallel Distrib. Comput.*, 1987, vol. 4, no. 5, pp. 439–458.
DOI: https://doi.org/10.1016/0743-7315(87)90018-9

[7] Kirk D.B., Hwu W.W. Programming massively parallel processors. Morgan Kaufmann, Elsevier, 2013.

[8] Izzo D., Rucinski M., Ampatzis C. Parallel global optimisation metaheuristics using an asynchronous island-model. *IEEE CEC'09*, 2009, pp. 2301–2308.
DOI: https://doi.org/10.1109/CEC.2009.4983227

[9] Lorion Y., Bogon T., Timm I.J., et al. An agent based parallel particle swarm optimization — APPSO. *IEEE Swarm Intelligence Symp.*, 2009, pp. 52–59.
DOI: https://doi.org/10.1109/SIS.2009.4937844

[10] Seliverstov E.Y., Karpenko A.P. Hierarchical model of parallel metaheuristic optimization algorithms. *Procedia Comput. Sc.,* 2019, vol. 150, pp. 441–449.
DOI: https://doi.org/10.1016/j.procs.2019.02.075

[11] Skillicorn D.B., Talia D. Models and languages for parallel computation. *ACM Comput. Surv.*, 1998, vol. 30, no. 2, pp. 123–169.
DOI: https://doi.org/10.1145/280277.280278

[12] Селиверстов Е.Ю. Графовые модели графического процессора. *Системы компьютерной математики и их приложения: Матер. XVIII Междунар. науч. конф.*, 2007, с. 117–119.

[13] Messmer B.T., Bunke H. Efficient subgraph isomorphism detection: a decomposition approach. *IEEE Trans. Knowl. Eng.*, 2000, vol. 12, no. 2, pp. 307–323.
DOI: https://doi.org/10.1109/69.842269

[14] Griva I., Nash S., Sofer A. Linear and nonlinear optimization. Philadelphia, SIAM, 2008.

[15] Li X., Tang K., Omidvar M.N., et. al. Benchmark functions for the CEC'2013 special session and competition on large-scale global optimization.
URL: http://www.tflsgo.org/assets/cec2018/cec2013-lsgo-benchmark-tech-report.pdf
(дата обращения: 10.09.2021).

[16] Kennedy J., Eberhart R. Swarm intelligence. Morgan Kaufmann, Elsevier, 2001.

[17] Parhami B. Introduction to parallel processing: algorithms and architectures. Nature Switzerland AG, Springer, 1999.

Селиверстов Евгений Юрьевич — аспирант кафедры «Системы автоматизированного проектирования» МГТУ им. Н.Э. Баумана (Российская Федерация, 105005, Москва, 2-я Бауманская ул., д. 5, стр. 1).

Просьба ссылаться на эту статью следующим образом:

Селиверстов Е.Ю. Структурное согласование алгоритмов глобальной оптимизации с архитектурой графических процессоров. *Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение*, 2022, № 2 (139), с. 42–59.
DOI: https://doi.org/10.18698/0236-3933-2022-2-42-59

STRUCTURAL MAPPING OF GLOBAL OPTIMIZATION ALGORITHMS TO GRAPHICS PROCESSING UNIT ARCHITECTURE

E.Yu. Seliverstov evgeny.seliverstov@omniverse.ru

Bauman Moscow State Technical University, Moscow, Russian Federation

Abstract

Graphics processing units (GPU) deliver a high execution efficiency for modern metaheuristic algorithms with a high computation complexity. It is crucial to have an optimal task mapping of the optimization algorithm to the parallel system architecture which strongly affects the efficiency of the optimization process. The paper proposes a novel task mapping algorithm of the parallel metaheuristic algorithm to the GPU architecture, describes problem statement for the mapping of algorithm graph model to the GPU model, and gives a formal definition of graph mapping and mapping restrictions. The algorithm graph model is a hierarchical graph model consisting of island parallel model and metaheuristic optimization algorithm model. A set of feasible mappings using mapping restrictions makes it possible to formalize GPU architecture and parallel model features. The structural mapping algorithm is based on cooperative solving of the optimization problem and the discrete optimization problem of the structural model mapping. The study outlines the parallel efficiency criteria which can be evaluated both experimentally and analytically to predict a model efficiency. The experimental section introduces the parallel optimization algorithm based on the proposed structural mapping algorithm. Experimental results for parallel efficiency comparison between parallel and sequential algorithms are presented and discussed

Keywords

Global optimization, task mapping, task scheduling, structural mapping, parallel systems, graphics processing units

Received 10.03.2022
Accepted 31.03.2022

REFERENCES

[1] Nvidia Ampere GA102 GPU architecture. *nvidia.com: website*.
Available at: https://www.nvidia.com/content/PDF/nvidia-ampere-ga-102-gpu - architecture-whitepaper-v2.pdf (accessed: 10.09.2021).

[2] Kwok Y.K., Ahmad I. Static scheduling algorithms for allocating directed task graphs to multiprocessors. *ACM Comput. Surv*., 1999, vol. 31, no. 4, pp. 406–471.
DOI: https://doi.org/10.1145/344588.344618

[3] Sinnen O. Task scheduling for parallel systems. New York, Wiley-Interscience, 2007.

[4] Hendrickson B., Kolda T.G. Graph partitioning models for parallel computing. *Parallel Comput*., 2000, vol. 26, no. 12, pp. 1519–1534.
DOI: https://doi.org/10.1016/S0167-8191(00)00048-X

[5] Chen S., Eshaghian M.M., Wu Y.C. Mapping arbitrary non-uniform task graphs onto arbitrary non-uniform system graphs. *Proc. Int. Conf. on Parallel Processing*, 1995, vol. 2, pp. 191–195.

[6] Berman F., Snyder L. On mapping parallel algorithms into parallel architectures. *J. Parallel Distrib. Comput*., 1987, vol. 4, no. 5, pp. 439–458.
DOI: https://doi.org/10.1016/0743-7315(87)90018-9

[7] Kirk D.B., Hwu W.W. Programming massively parallel processors. Morgan Kaufmann, Elsevier, 2013.

[8] Izzo D., Rucinski M., Ampatzis C. Parallel global optimisation meta-heuristics using an asynchronous island-model. *IEEE CEC'09*, 2009, pp. 2301–2308.

DOI: https://doi.org/10.1109/CEC.2009.4983227

[9] Lorion Y., Bogon T., Timm I.J., et al. An agent based parallel particle swarm optimization — APPSO. *IEEE Swarm Intelligence Symp*., 2009, pp. 52–59.
DOI: https://doi.org/10.1109/SIS.2009.4937844

[10] Seliverstov E.Y., Karpenko A.P. Hierarchical model of parallel metaheuristic optimization algorithms. *Procedia Comput. Sc*., 2019, vol. 150, pp. 441–449.
DOI: https://doi.org/10.1016/j.procs.2019.02.075

[11] Skillicorn D.B., Talia D. Models and languages for parallel computation. *ACM Comput. Surv*., 1998, vol. 30, no. 2, pp. 123–169.
DOI: https://doi.org/10.1145/280277.280278

[12] Seliverstov E.Yu. [Graph models of GPU]. *Sistemy komp'yuternoy matematiki i ikh prilozheniya. Mater. XVIII Mezhdunar. nauch. konf.* [Systems of Computer Maths and their Application. Proc. XVIII Int. Sc. Conf.], 2007, pp. 117–119 (in Russ.).

[13] Messmer B.T., Bunke H. Efficient subgraph isomorphism detection: a decomposition approach. *IEEE Trans. Knowl. Data Eng*., 2000, vol. 12, no. 2, pp. 307–323.
DOI: https://doi.org/10.1109/69.842269

[14] Griva I., Nash S., Sofer A. Linear and nonlinear optimization. Philadelphia, SIAM, 2008.

[15] Li X., Tang K., Omidvar M.N., et. al. Benchmark functions for the CEC'2013 special session and competition on large-scale global optimization.
Available at: http://www.tflsgo.org/assets/cec2018/cec2013-lsgo-benchmark-tech-report.pdf (accessed: 10.09.2021).

[16] Kennedy J., Eberhart R. Swarm intelligence. Morgan Kaufmann, Elsevier, 2001.

[17] Parhami B. Introduction to parallel processing. Nature Switzerland AG, Springer, 1999.

Seliverstov E.Yu. — Post-Graduate Student, Department of Computer-Aided Design Systems, Bauman Moscow State Technical University (2-ya Baumanskaya ul. 5, str. 1, Moscow, 105005 Russian Federation).

Please cite this article in English as:

Seliverstov E.Yu. Structural mapping of global optimization algorithms to graphics processing unit architecture. *Herald of the Bauman Moscow State Technical University, Series Instrument Engineering*, 2022, no. 2 (139), pp. 42–59 (in Russ.).
DOI: https://doi.org/10.18698/0236-3933-2022-2-42-59